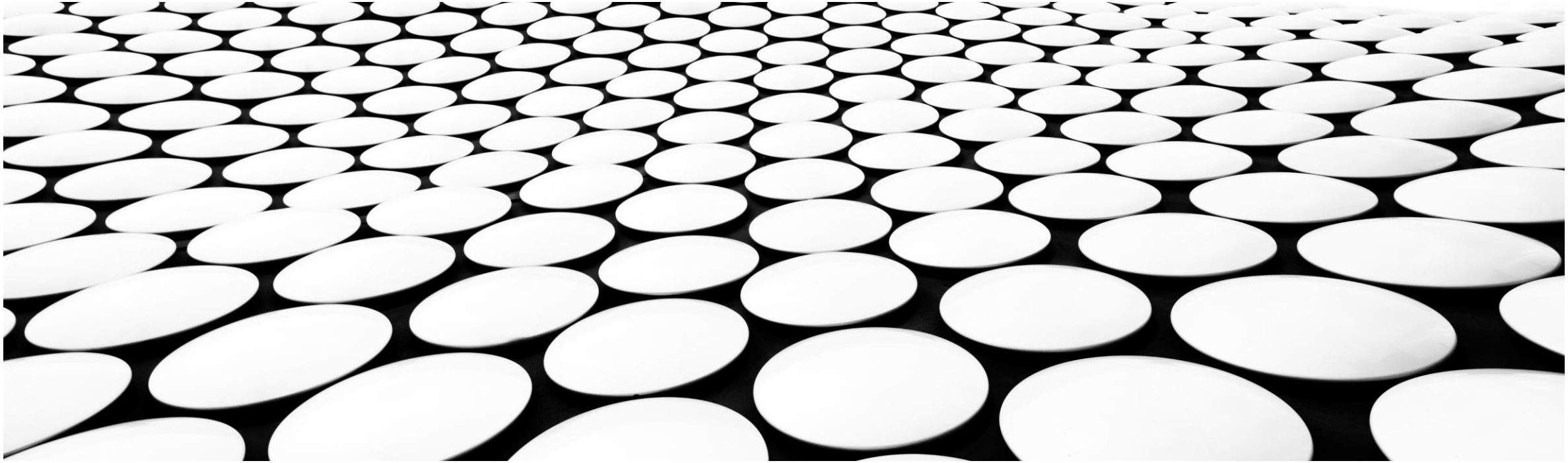
Building A Distributed Systems Pipeline: An Overview

COEN-317: Distributed Systems
Robert Bruce
Department of Computer Science and Engineering
Santa Clara University



Elements of Distributed Systems

Master queue machine:

A dedicated, network-connected computer which accepts incoming socket connections to process new jobs and monitor existing jobs distributed to worker machines

Master queue machine responsibilities:

- Dividing new job requests into a series of parallel, concurrent tasks based on the number of currently available (idle) worker machines.
- Reallocating job queue on worker machines that are idle (load balancing).
- Monitoring worker machine failures and reallocating job queues to other machines in such situations (for resiliency and reliability of job completion).

Elements of Distributed Systems

Worker machine

- A dedicated, network-connected computer which listens on a socket connection for incoming job (task) requests from the master queue.
- There are typically many worker machines.

The queue system which runs on worker machine:

- Runs multiple POSIX threads (pthreads):
    1. Boss thread launches multiple worker threads depending on number of cores in physical processor.
    2. Worker thread listens for incoming connection requests from Master queue machine.
    3. One or more threads idle in wait-loops for task requests.

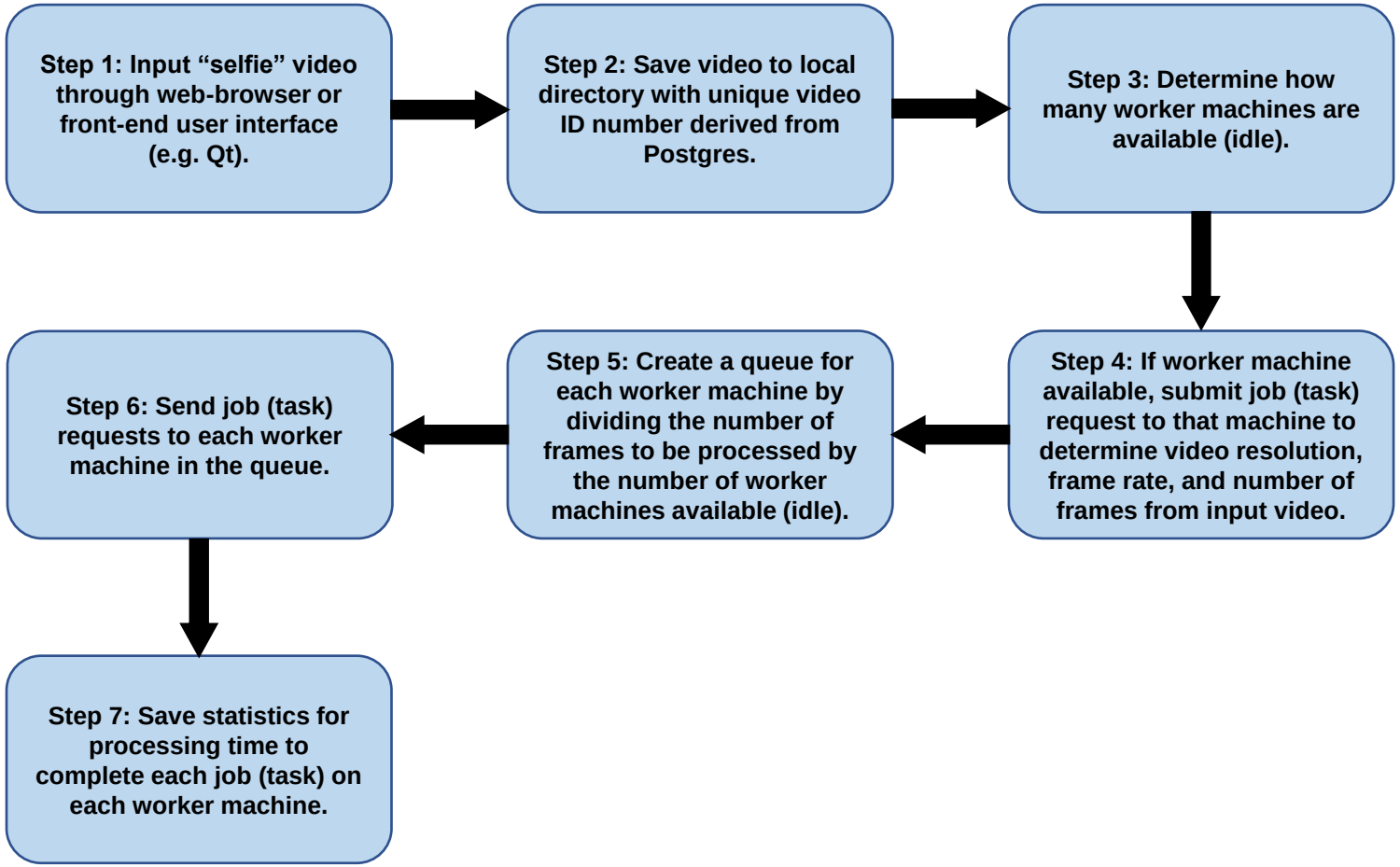
Elements of Distributed Systems

A database may be used to store vital information to effectively manage the distributed system.

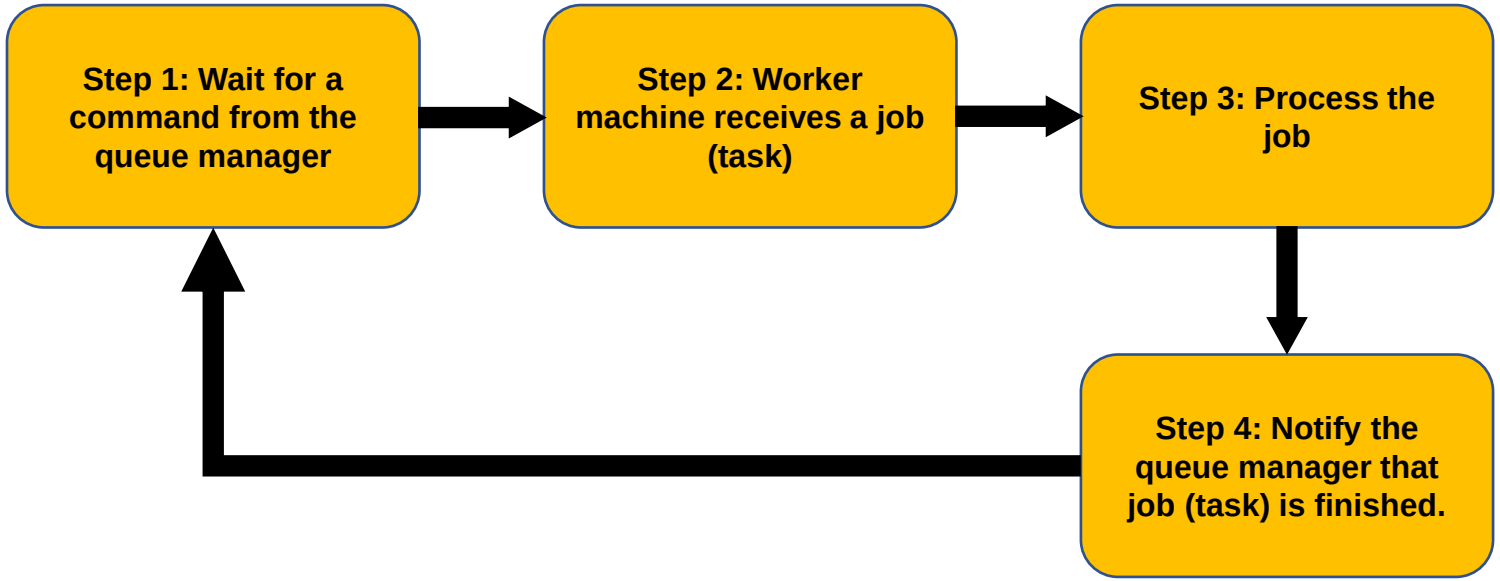
Database Management System (DBMS) information stored:

- Worker machine hardware profile
    1. IP address
    2. number of CPU cores
    3. RAM size
    4. available local disk storage
- Current state of worker machines
    1. idle,
    2. processing
    3. inaccessible
- Average processing time per job (task) on each worker machine
- System log (syslog) information from worker machines

Master queue machine



Worker machine



Message queue data structure

One function of the master queue machine is maintaining a message queue: a series of messages from worker machines.

- These messages are typically added FIFO (first-in-first-out) order in a data structure such as a linked list.
- I recommended using a null-terminated linked-list when first implementing your project.
- A more advanced technique would be to implement a large ring buffer instead of a null-terminated linked-list.

Commands needed for communication: master queue and worker machine

For a minimum viable project in this class, you will need the following commands and appropriate responses:

- **view job** : This command is issued by the master queue machine to a worker machine to determine what (if any) jobs (or tasks) are currently being processed by a specific worker machine. Possible responses from worker machine: “no job” or “processing *<video_id> <frame>*” where *<video_id>* is a unique video identification number and *<frame>* is a given frame from that video.
- **put file *<filename>* with length *<non-negative integer>* *<contents of file>*** : This command is issued by the master queue machine to a worker machine to upload an image file of specified length (in bytes) with given filename. Response from worker machine should be: “received file *<filename>* with length *<non-negative integer>*”.
- **get file *<filename>*** : This command is issued by the master queue machine to copy a file from the worker machine to the master queue machine. Response from worker machine should be one of the following: “retrieve file not found for file *<filename>*” (if the file is non-existent on local worker machine) or “sent file *<filename>* with length *<non-negative integer> <contents of file>*”.
- **get video metadata for file *<filename>*** : This command is issued by the master queue machine to a worker machine to determine metadata information about an input video specified by <filename>. The metadata minimally should include frame rate in frames per second (25, 29.97, 30, 60, etc.), length of video (in seconds), and video resolution (in pixels). Response from worker machine should be one of the following: “unknown video format for file *<filename>*” (if file is an unknown format that FFMPEG cannot decode) or “video metadata for file *<filename>* framerate *<frames per second>*, duration *<length of video in seconds>*, resolution *<x,y>*”.
- **extract video images with framerate *<frames per second>* duration *<length of video in seconds>* for file *<filename>*** : This command is issued by the master queue machine to a worker machine to extract still images from a video input file on the local worker machine’s temporary directory. Response from worker machine should be one of the following: “extract video failed: unknown video format for file *<filename>*” (if file is an unknown format that FFMPEG cannot decode), “extract video failed: local disk storage full” (if there is no local storage to create image files from video), or “video extracted for file *<filename>* with file sequence *<filename$_1$> <filename$_2$> <filename$_3$> … <filename$_n$>* ”.
- **create video from images with framerate *<frames per second>* , duration *<length of video in seconds>*, for file sequence *<filename$_1$> <filename$_2$> <filename$_3$> … <filename$_n$>*** : This command is issued by the master queue machine to a worker machine to convert *N* image files into a video (where *N* is the number of images). You may use any video compression format you prefer. Response from worker machine should be one of the following: “failed to create video” (if a failure occurs) or “video created with filename *<filename>*”.
- **detect face for file *<filename>*** : This command is issued by the master queue machine to a worker machine to use dLib to determine the location (if any) of a face in the image file. If multiple faces are detected, return the first one found by dLib and ignore remaining faces in the image. Response from worker machine should be: “face detected for file *<filename>* with face data points *(x_1,y_1), (x_2,y_2), (x_3,y_3),…(x_{68},y_{68})*”. If no faces are detected, return “no face detected for file *<filename>*”
- **draw mesh for file *<filename>* with face data points *(x_1,y_1), (x_2,y_2), (x_3,y_3),…(x_{68},y_{68})*** : This command is issued by the master queue machine to a worker machine to use OpenCV to compute then draw Delaunay triangles (given the sixty-eight Cartesian coordinate inputs) on the image file specified by *<filename>*. Responses form worker machine should be: “file not found for file *<mesh filename>* (if file does not exist) or “mesh drawn for file *<mesh filename>*”.
- **delete *<filename>*** : This command is issued by the master queue machine to a worker machine to delete *<filename>* from the worker machine’s temporary file area. Responses form worker machine should be: “file not found for file not found on local machine or “deleted file *<filename>*”.
- **delete all files** : This command is issued by the master queue machine to a worker machine to delete all old and unused files in the worker machine’s temporary file area. Response from worker machine should be: “temporary directory empty” (if directory was empty or if temporary directory files were successfully deleted on worker machine temporary file area). An error should be sent back if temporary files are unable to be deleted.

Example transaction between master queue machine and worker machine

IP address:

Master queue machine: 192.168.1.10

Worker machine: 192.168.2.25

Example communications:

192.168.1.10: view job

192.168.2.25: <Jan 12, 2023 04:18:23 PM> no job

192.168.1.10: put file 112.2.png with length 19349 *binary contents of 112.2.png*

192.168.2.25: <Jan 12, 2023 04:22:12 PM> received file 112.2.png with length 19349

192.168.1.10: detect face for file 112.2.png

192.168.2.25: <Jan 12, 2023 04:24:31 PM> face detected for file 112.2.png with face data points (100,338), (105,324), (106,305), …

192.168.1.10: draw mesh for file 112.2.png with face data points (100,338), (105,324), (106,305), …

192.168.2.25: <Jan 12, 2023 04:26:42 PM> mesh drawn for file mesh-112.2.png

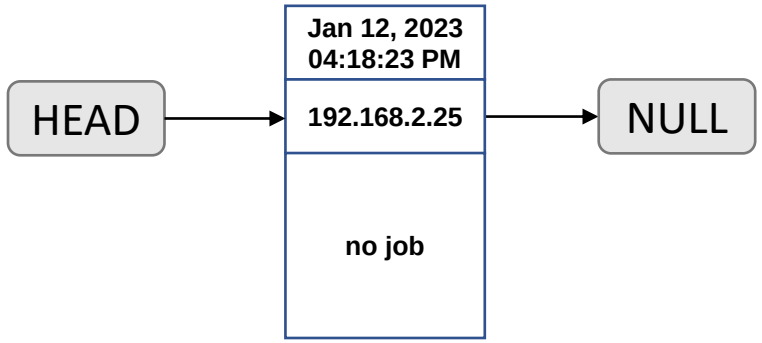
192.168.1.10: get file mesh-112.2.png

192.168.2.25: <Jan 12, 2023 04:27:01 PM> sent file mesh-112.2.png with length 20010 *binary contents of mesh-112.2.png*

Message queue data structure

192.168.1.10: view job

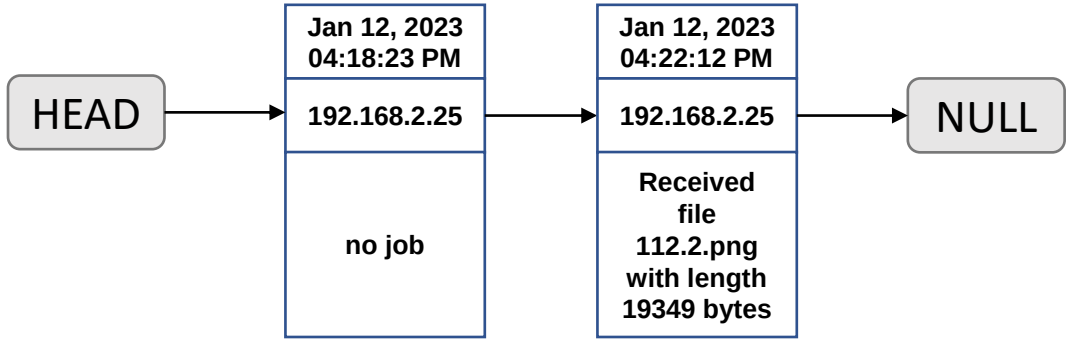
192.168.2.25: <Jan 12, 2023 04:18:23 PM> no job



Message queue data structure

192.168.1.10: put file 112.2.png with length 19349 ***binary contents of 112.2.png***

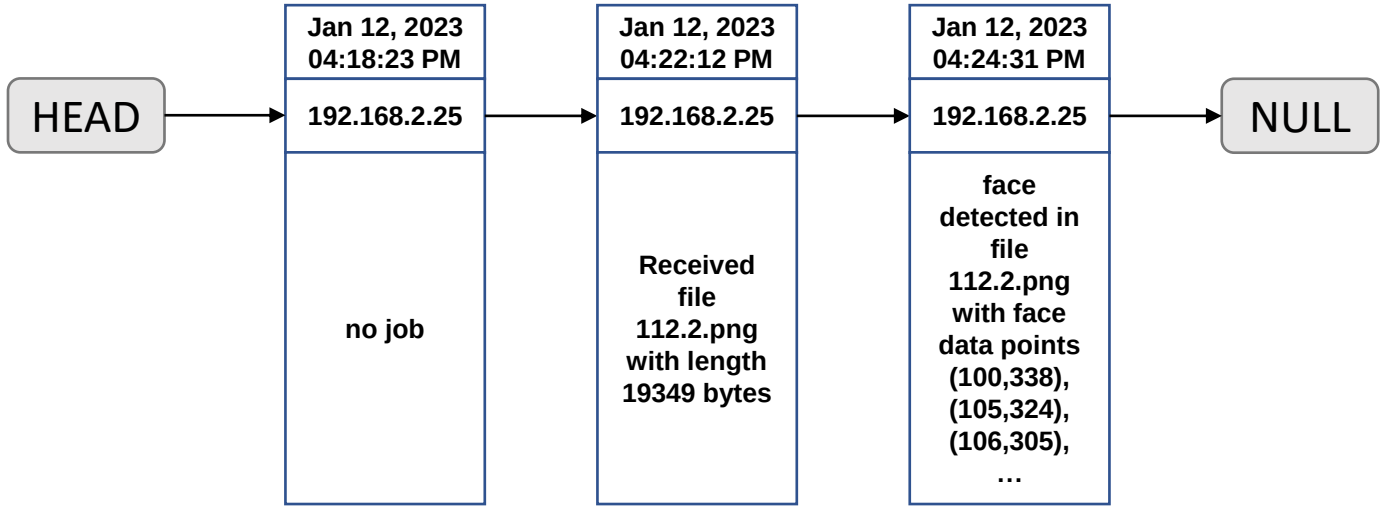
192.168.2.25: <Jan 12, 2023 04:22:12 PM> received file 112.2.png with length 19349 bytes



Message queue data structure

192.168.1.10: detect face for file 112.2.png

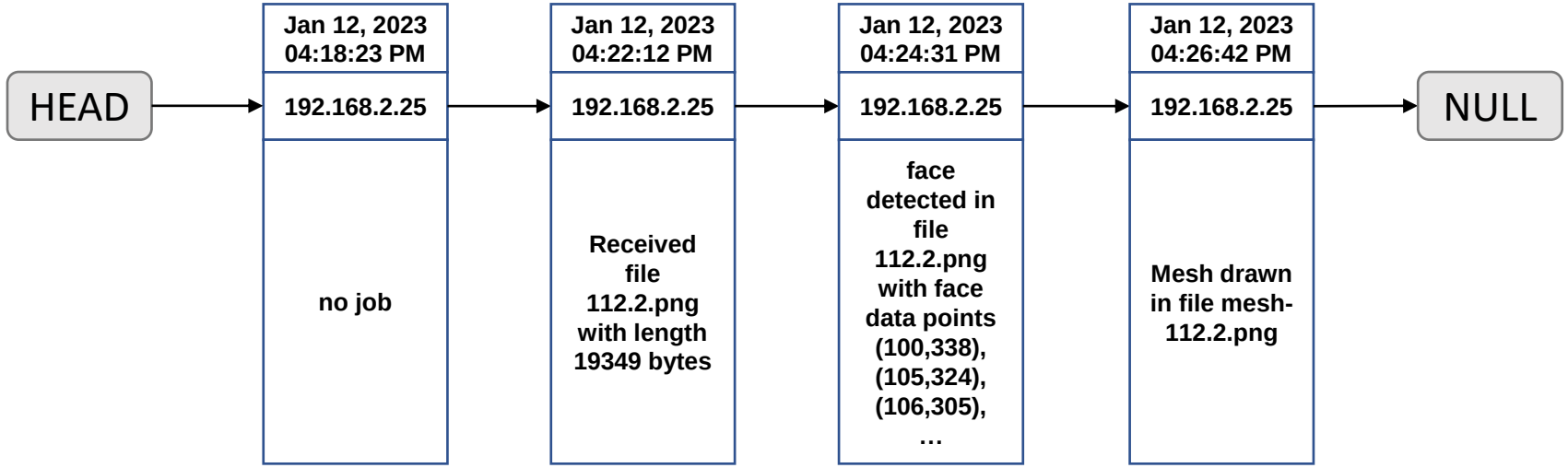
192.168.2.25: <Jan 12, 2023 04:24:31 PM> face detected in file 112.2.png with face data points (100,338), (105,324), (106,305), …



Message queue data structure

192.168.1.10: draw mesh for file 112.2.png with face data points (100,338), (105,324), (106,305), …

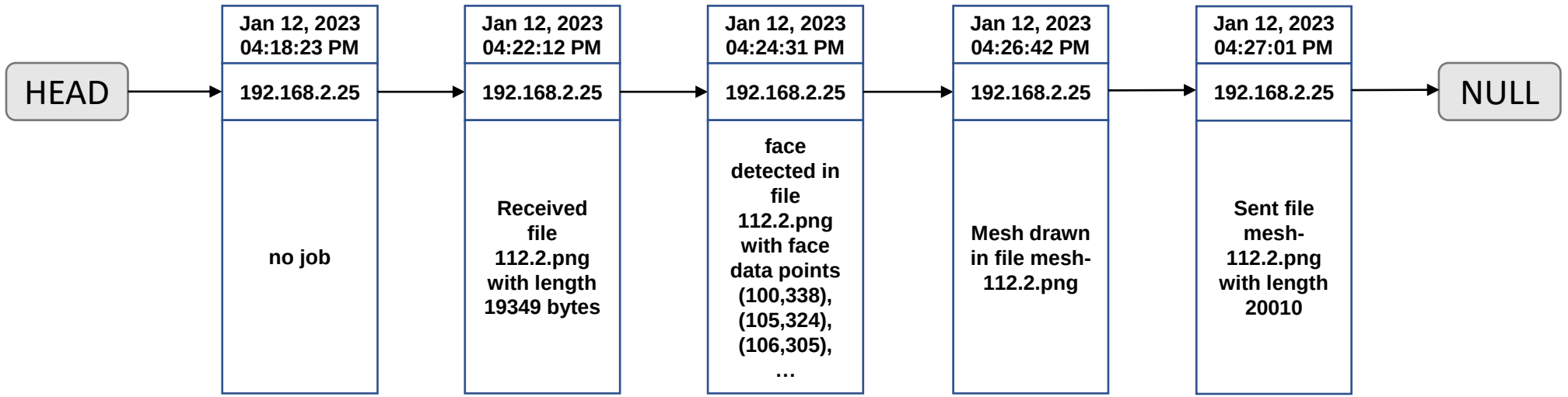
192.168.2.25: <Jan 12, 2023 04:26:42 PM> mesh drawn in file mesh-112.2.png



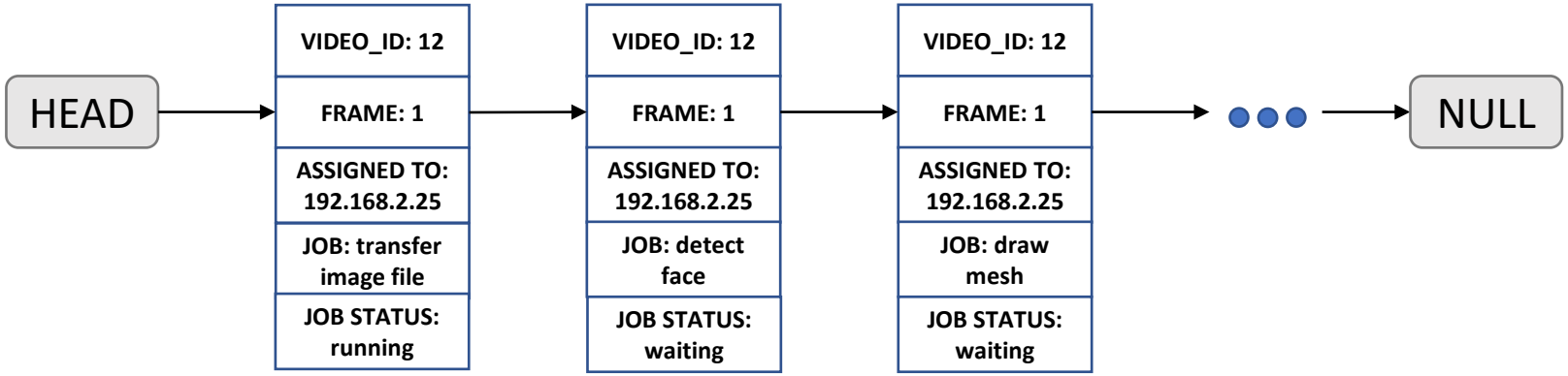
Message queue data structure

192.168.1.10: get file mesh-112.2.png

192.168.2.25: <Jan 12, 2023 04:27:01 PM> sent file mesh-112.2.png with length 20010 ***binary contents of mesh-112.2.png***



Job queue data structure



Implementing the queue: some suggestions

1. Start with simple socket communication between master queue machine and one worker machine. The master queue machine and worker machine may even be the same physical machine.
2. Focus on parsing these messages on both the Master queue and worker machine. Parsing will enable you to extract filename information, and file size information.
3. Once parsing in step 2 is completed, see if you can successfully send a binary file between master queue machine and worker machine. The worker machine should save the file to a local working directory (a temporary working directory) on the worker machine.
4. Once step 3 is completed, see if you can successfully purge local temporary files (if any) on worker machine by issuing a purge command from master queue machine.
5. Once steps 1 through 4 are completed, see if you can begin implementing pthreads on the Master queue machine to begin processing messages on two or more worker machines concurrently.

TEAM BUILDING EXERCISE

Once teams are established, please begin discussing roles and responsibilities on your team.

- What will be the roles of each team member?
- Team meetings: how will you communicate (in-person, email, voice-over-IP, etc.)?
- Team meetings: what time will you conduct team meetings? What frequency (i.e. daily, weekly, monthly)?
- How will your team resolve conflict in the event of disagreement over implementation details?